SAP HANA Data Modeling Tools Overview

SAP HANA Information Modeling which also known as **SAP HANA Data Modeling** is the heart of HANA application development.

Since SAP HANA is a radically new database "underneath the hood," SAP had to provide DBAs, data architects and others a familiar way to interact with the tables while maintaining a level of abstraction to ensure that people wouldn't disrupt the tables. That's how you get the "virtual" data modeling capabilities of SAP HANA. Although people can log in and "see" the tables, they aren't really there, like they are in a traditional disk-based database. What you're seeing is a virtual representation of the tables since the actual tables aren't physically persisted on the storage medium as they would be in a disk-based database. This virtual data model allows people incredible flexibility when manipulating the data and protects them from some of the more nasty effects of playing around with physical tables in the database.

In the context of SAP HANA, data modeling can be viewed as the construction of different types of views of the data tables maintained in the SAP HANA database. Modeling defines how you are going to access the data that's physically stored in HANA tables. Views can be thought of as "virtual" tables that are built up from underlying data structures in memory or from other views.

From an SAP HANA perspective, data modeling defines how you're going to store and access your data. By creating views, you build new layers of access to your data that are derived from what's in physical RAM storage, but that is calculated or adapted based on your application needs. The views are built on demand and are always up to date, and they can contain complex calculations that are computed within the database.

Because SAP HANA is a fast, in-memory database, you can build virtual models that are more flexible and powerful than those found in typical disk-based database designs. HANA is optimized for aggregating mass data on the fly, and thus it allows you to build models on top of raw transaction data without first doing pre-aggregation or creating materialized views.

The concept of a logical view in a database is pretty universal. The logical structure points to where the physical data is stored. What's different in HANA is that operations on large quantities of data are so fast that it's not necessary to build a persistent additional physical view or an additional index on the data to make it go fast. Rather than building redundant tables for speed, you aggregate data on the fly. The key in there is that HANA is not re-persisting the data multiple times for every different view of data that we want to do. It is stored once and then we can create a lot of different logical views at that point to the data that is physically in the database for different use cases and different application uses without having to make copies of the data to support additional views.

Another upside to logical views: Extending data modeling to more stakeholders.

The great thing about creating logical views of the data is that you can allow more people to create those views because their views don't change anything about the underlying data store. You might never allow less technical people to access a traditional database, but with the ability to create logical views of the data that don't change the underlying data store (let alone corrupt it), you can allow as many people as are interested to get involved with modeling in SAP HANA.

With HANA, you're not modeling to get around disk space constraints, and you don't need to model and keep in mind where your data's partitioned and where it's coming from and how to best access it from different storage pieces to reduce the lag time of disks. With HANA you can create queries that are more complex and still achieve high performance using straightforward SQL statements. If you're coming from a legacy system with maybe a hundred different types of models that draw on different master data, you may find that you need only a tenth that many in HANA because you are no longer having to design models with regard to disk or data volume constraints. You can blend models together and get a broader view of what's going on with the data, with more granularity than you could before.

Becoming proficient at data modeling for HANA is one of the key elements of extracting all of the performance out of the system. Understanding how the system works, where its speed advantages are, and how to get the best performance from the system are all best done with hands-on experience.

Data Modeling Tools

The **SAP HANA STUDIO** is used for creating data models within SAP HANA. It can be used to explore and analyze existing data models, make modifications, and build new data models from scratch. The modeler generally uses the Administration Console and Modeler perspectives and tools of SAP HANA Studio.

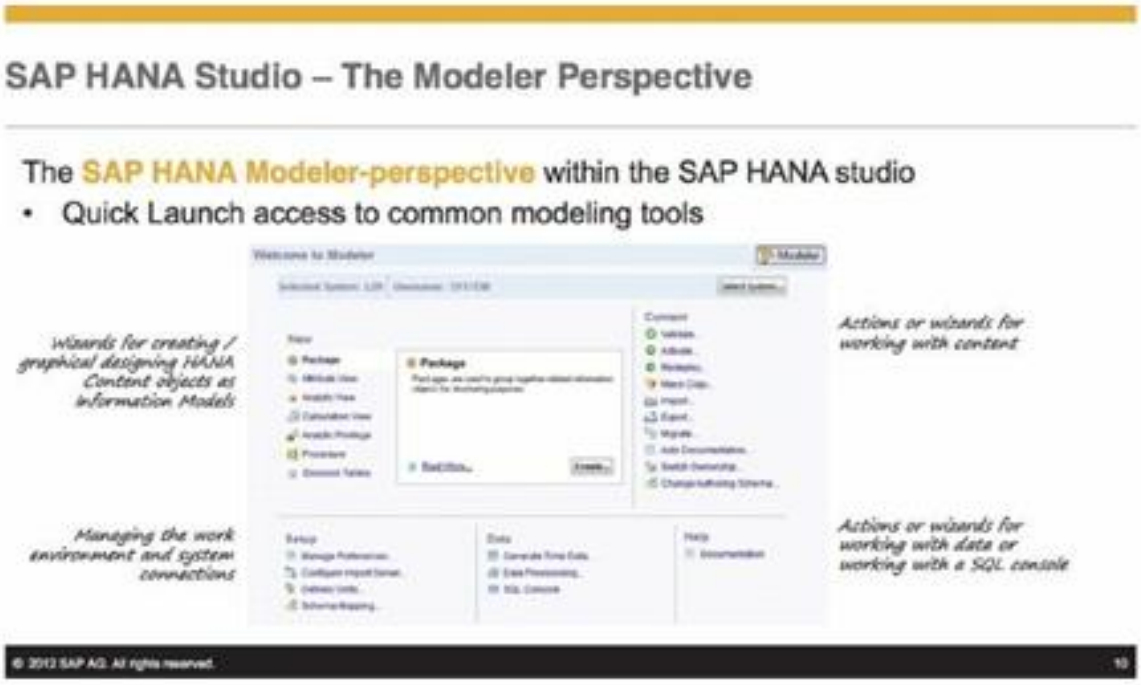


Figure 6-1. SAP HANA Studio: Modeler Perspective

Packaged applications for SAP HANA come with their own data models. You can inspect them with SAP HANA Studio and tailor them to your needs.

BI clients use data models directly for reports. Views that are created inside SAP HANA can be treated by BI tools as any other table, allowing for direct access for reporting. If you calculate net sales in your data model and have the HANA engine execute the calculation of net sales in memory and then just send the result set up to your query tool, it's much more efficient than having the query tool pull up all the raw data and then having the BI query execute what the net sales difference is.

Application developers use the output of data models as inputs into their applications. For best performance, application developers will want to maximize the number of calculations that are done in the database and to reduce the amount of data that's transferred back up to the application. It, in turn, allows the application developer to offload the bulk of the calculation logic into the database, making it possible for application logic to be greatly simplified and for views to be useful across a range of tools.